DESARROLLO DE UN MANUAL SOBRE PRUEBAS DE SOFTWARE DURANTE EL CICLO DE DESARROLLO E IMPLEMENTACIÓN

PROYECTO DE GRADO

Requisito final para obtener el título de Ingeniero de Sistemas y Computación

Presentado por:

LUISA MARIA ZULUAGA JARAMILLO

YISETH MELISSA BEDOYA CARDONA

UNIVERSIDAD TECNOLÓGICA DE PEREIRA

FACULTAD DE INGENIERÍAS: EEFCC

INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

PEREIRA, SEPTIEMBRE 2018

DESARROLLO DE UN MANUAL SOBRE PRUEBAS DE SOFTWARE DURANTE EL CICLO DE DESARROLLO E IMPLEMENTACIÓN

PROYECTO DE GRADO

Requisito final para obtener el título de Ingeniero de Sistemas y Computación

Presentado por:

LUISA MARIA ZULUAGA JARAMILLO

YISETH MELISSA BEDOYA CARDONA

Director:

MSC. CÉSAR JARAMILLO

UNIVERSIDAD TECNOLÓGICA DE PEREIRA

FACULTAD DE INGENIERÍAS: EEFCC

INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

PEREIRA, SEPTIEMBRE 2018

Contenido

Lista de Tablas

Lista de Ilustraciones

Lista de Anexos

Introducción

Este documento tiene como función describir y ordenar los principales conceptos necesarios para realizar pruebas de software. Se busca describir los tipos de pruebas que existen y sus objetivos, las diferentes funcionalidades que tienen cada una, así como su importancia durante el proceso. Adicional clarificar las diferentes metodologías que existen para realizar pruebas de software y cómo aplicarlas según el tipo de proceso realizado durante la producción o modificación del software.

Las pruebas de software o testing constituyen las validaciones que se le deben realizar a un software al momento de producirse o modificarse. Estas pruebas lo que buscan es detectar fallas en el sistema que ocasionen un mal funcionamiento de este. Para realizar este proceso de pruebas

se utilizan diferentes métodos y estrategias con el fin de realizar las entradas al sistema y analizar las salidas y determinar si estas cumplen con el objetivo propuesto.

Debido a lo anterior, se puede afirmar que identificar, ordenar y describir algunos de los diferentes tipos de pruebas de software dentro de un ciclo de desarrollo, puede ser una tarea tan compleja como el mismo oficio de realizarlas, no solo por tener claras las técnicas a utilizar es garantía que el producto será exitoso, es decir, que cumplirá con las expectativas del sistema brindando a los usuarios las funcionalidades para las cuales fue creado o modificado. Es necesario acompañar este proceso de actividades de gran valor como entender por parte del testing la función del producto y su finalidad para el consumidor. El probador debe estar capacitado además de técnicas claras para la detección de errores y una gran calidad para hallar sus soluciones

Muchas de las empresas que desarrollan software cumplen con la tarea de realizar las pruebas de sus productos, estas pueden ser tan rigurosas entendiendo este concepto como hacer verificaciones utilizando pruebas como de seguridad, integridad, funcionalidad la cual está dada por las exigencias del usuario final y la organización, entre otras, así como tan básicas donde solo se realizan pruebas con el desarrollador verificando que la funcionalidad general cumpla con las expectativas del usuario sin revisar a fondo el impacto que tiene el desarrollo en todo el sistema. Definir la técnica a utilizar, su entorno, del mismo modo tener el personal adecuado y capacitado es una tarea importante, pero poco valorada usualmente por el cliente final.

I. Marco de Referencia

Para el desarrollo del proyecto se necesitará tener claros los siguientes conceptos:

- Manual: Un manual es un libro o folleto en el cual se recogen los aspectos básicos, esenciales de una materia. Así, los manuales nos permiten comprender mejor el funcionamiento de algo, o acceder, de manera ordenada y concisa, al conocimiento algún tema o materia.
- Pruebas de Software: Las pruebas de software (Software Testing) comprenden el conjunto de actividades que se realizan para identificar posibles fallos de funcionamiento, configuración o usabilidad de un programa o aplicación, por medio de pruebas sobre el comportamiento de este.
- Ciclo de desarrollo: Es todo el conjunto de actividades que se deben realizar para desarrollar un software. Estas actividades se clasifican en levantamiento de requerimientos, diseño, pruebas, implementación, verificación y mantenimiento,
- Ciclo de implementación: Proceso que comprende la realización de pruebas se software, verificación del funcionamiento del sistema, correcciones de desarrollo y planeación de la puesta en productivo del sistema desarrollado.
- Tester: Persona encargada de realizar las pruebas de software.
- Técnica: Conjunto de procedimientos o recursos que se usan en un arte, en una ciencia o en una actividad determinada, en especial cuando se adquieren por medio de su práctica y requieren habilidad.

- Metodologías: El término metodología se define como el grupo de mecanismos o procedimientos racionales, empleados para el logro de un objetivo, o serie de objetivos que dirige una investigación científica. Este término se encuentra vinculado directamente con la ciencia, sin embargo, la metodología puede presentarse en otras áreas como la educativa, en donde se encuentra la metodología didáctica o la jurídica en el derecho.
- Software: Se considera que el software es el equipamiento lógico e intangible de un ordenador. En otras palabras, el concepto de software abarca a todas las aplicaciones informáticas, como los procesadores de textos, las planillas de cálculo y los editores de imágenes.
- Requerimientos de software: Es una descripción completa del comportamiento del sistema que se va a desarrollar. Incluye un conjunto de casos de uso que describe todas las interacciones que tendrán los usuarios con el software
- Error: Acción humana que producto un resultado incorrecto
- Defecto: Desperfecto en un componente o sistema que ocasiona que no se comporte de la manera esperada
- Fallo: Manifestación física o funcional de un defecto
- Sistema: Un sistema es un conjunto de funciones que operan en armonía o con un mismo propósito, y que puede ser ideal o real

II. Antecedentes

En el desarrollo de la investigación para la realización de este trabajo, se encuentra que en la actualidad se utilizan muchas aplicaciones que facilitan la vida del ser humano, y en múltiples situaciones es importante el correcto funcionamiento de estas en los dispositivos tecnológicos,

un ejemplo de ellos son la telemedicina, movimientos bancarios, comunicaciones, transporte aéreo, entre otros, donde se observa la importancia de la tecnología y la calidad en el desarrollo de las aplicaciones.

Es de vital importancia tener software de calidad, ya que las fallas que puedan tener pueden provocar pérdidas humanas y perdida de dinero en las organizaciones, algunos ejemplos de fallas de software son:

“sonda espacial MCO (Mars Climate Orbiter) lanzada en 1998 y cuyo objetivo era rastrear el clima de Marte y transmitir los datos obtenidos a la Tierra; sin embargo, justo al realizar las maniobras de inserción a la órbita marciana, se perdió toda comunicación con el satélite. La junta de investigación de accidentes determinó que el origen de esta millonaria pérdida fue debido a que los datos generados por el programa que calculaba el rendimiento del propulsor estaban en unidades inglesas y que, a su vez, los resultados de este proceso eran utilizados por otro módulo que determinaba la desaturación del movimiento angular, pero los requería en unidades MKS. Esta incompatibilidad de unidades originó pequeños errores traducidos en una trayectoria seguida por la nave 170 kms más baja que la planeada, dando como resultado la destrucción del orbitador en la atmósfera de Marte

Asimismo, durante 2003, el noreste y medio oeste de Estados Unidos, así como la provincia de Ontario en Canadá, sufrieron un apagón de enormes dimensiones que duró alrededor de 7 horas y en otros lugares hasta una semana, afectando a más de 55 millones de personas. Una de las causas que contribuyeron a este problema fue un error de software. Semanas después de haberse

presentado el percance, los ingenieros de GeneralElectric realizaron una auditoría a su sistema. Después de analizar varios millones de líneas de código, detectaron que un error en la programación provocó una falla en el sistema de alarma del centro de control de energía de la empresa FirstEnergy propiciando con ello el apagón.

En años más recientes los errores de software no han dejado de existir. Durante 2014 el Banco Real de Escocia fue multado con 56 millones de libras esterlinas luego que, por un error de software, fueran cerradas durante varias semanas las cuentas de sus más de 6.5 millones de clientes impidiendo realizar operaciones bancarias. Por otro lado en 2015, Nissan hizo un llamado a más de un millón de sus vehículos debido a que identificó problemas de software vinculados con las bolsas de aire pues el programa no detectaba la presencia de un adulto ubicado en el asiento del acompañante. También en diciembre de ese año, el espacio aéreo londinense fue cerrado por un malfuncionamiento del software encargado de rastrear y planear el despegue y aterrizaje de aviones de uno de los aeropuertos más congestionados del mundo ocasionando que cientos de vuelos fueran cancelados, demorados o desviados."

Fuente:
http://www.iingen.unam.mx/esmx/Publicaciones/GacetaElectronica/Agosto2016/Paginas/Erroresdesoftware.aspx

Es por esto que durante proceso de creación o modificación de un software se evidencia que hay un conjunto de actividades el cual se denomina pruebas de software o testing el cual está encargado de realizar las validaciones con el fin de garantizar que el sistema cumpla con los objetivos propuestos.

De los diferentes procesos que se realizan en el desarrollo de software, las pruebas de software cobran importancia al entender que a través de ellas se garantiza que el sistema cumpla con los requerimientos funcionales y no funcionales exigidos, se realicen las validaciones e impactos que tendrá al momento de su implementación con los diferentes softwares con los que interactúa y adicional prevenir posibles errores en el desarrollo de las actividades de los funcionarios.

Para realizar el proceso de pruebas de software es necesario contar con una formación académica la cual brinde las competencias necesarias para identificar cuáles son los tipos de pruebas que existe, sus utilidades y los momentos en los cuales se deben aplicar. Así como tener presente las diferentes metodologías que existen con el fin de analizar y determinar cuál es la mejor opción para el proyecto en el cual se está trabajando.

En este trabajo se hace referencia a diferentes trabajos de investigación o proyectos que se han sido propuestos o se piensan desarrollar y sirvieron como insumo por su relación con lo propuesto en esta investigación:

En el ámbito internacional encontramos un documento sobre “Pruebas de Software, Fundamentos y Técnicas” de la Universidad Politécnica de Madrid, este trabajo describe las herramientas y las técnicas necesarias para el proceso de pruebas. En Uruguay en la Universidad de la Republica Beatriz Pérez desarrollo para su maestría una tesis sobre “El Proceso de Testing Funcional e Independiente”, el proceso que se definió tiene por nombre ProTest.

En ámbito nacional podemos destacar un “Estudio sobre realización y documentación de pruebas software” que se realizó en el año 2015 en la Universidad industrial de Santander, donde describen el proceso realizado y resultados obtenidos al estudiar la documentación técnica de los desarrollos de software del grupo de investigación en sistemas y tecnologías de la información de

dicha universidad, adicional tenemos una tesis de grado de la Universidad Nacional de Colombia – sede Medellín, este investigación es una “Propuesta Metodológica para la Realización de Pruebas de Software en Ambientes Productivos”

En el ámbito local encontramos La tesis “Estudio de las prácticas de calidad del software implementadas en las Mipymes desarrolladoras de software de Pereira”, de las estudiantes Paola Andrea Ramírez Aguirre y Carolina Ramírez Arias, para optar por el título de Ingenieras de Sistemas de la Universidad Tecnológica de Pereira, y se fundamenta en la siguiente pregunta: “¿Cuál es el grado de implementación de modelos de calidad de software en las empresas desarrolladoras de la ciudad de Pereira?”

III. Planteamiento del Problema

En un proyecto de software se puede encontrar errores a lo largo del ciclo del desarrollo y cuando se entrega el producto final, es por esto, por lo que se encuentra la importancia que tienen las pruebas de software en el ciclo del desarrollo, se debe garantizar la realización de estas, a pesar de que puede ser costoso el no realizarlas y encontrar errores después de la implementación puede ser aún más costoso.

Obtener información, manuales o guías sobre pruebas de software es un proceso difícil, debido a que no se cuenta con mucha información al respecto y menos con un enfoque académico que motive a los estudiantes a realizar dichas pruebas o enfocarse por esta rama tecnológica, es por esto, que se ha decidido realizar un manual de fácil acceso y con información clara que permita entender y aplicar las pruebas durante el ciclo de desarrollo de forma correcta y ágil.

Se realiza una investigación sobre manuales o guías para el desarrollo de software se y se ha evidenciado que la información al respecto es muy básica teniendo en cuenta que el proceso de pruebas es de suma importancia para el desarrollo de una implementación.

Por lo anterior se evidencia una deficiencia en la información disponible para los tester de software debido a que se encuentra información sobre la definición y conceptos adicionales del área, pero no se cuenta con un manual que sirva como guía para entender cómo aplicar los diferentes tipos de pruebas y su objetivo.

Para subsanar esta problemática se decide realizar el desarrollo del presente proyecto.

IV. Objetivos

A. Objetivo General

Desarrollar un manual para aplicar pruebas de software durante el ciclo de desarrollo y el proceso de implementación.

B. Objetivos Específicos

1. Especificar la forma de aplicar pruebas de software en todo el ciclo del desarrollo del proyecto para crear software de calidad.
2. Explicar las diferentes metodologías para aplicar pruebas de software en el ciclo de desarrollo del proyecto buscando evitar fallos y defectos que retrasen el desarrollo del proyecto y por ende aumente su costo.
3. Construir un manual que sea de fácil acceso y claro para que los estudiantes aprendan a utilizar las pruebas de software en el ciclo de desarrollo de un proceso de implementación

V. Justificación

Durante el proceso de creación o modificación de un software se evidencia que hay un proceso el cual se denomina pruebas de software o testing el cual está encargado de realizar las validaciones con el fin de garantizar que el sistema cumpla con los objetivos propuestos.

De los diferentes procesos que se realizan en el desarrollo de software, las pruebas de software cobran importancia al entender que a través de ellas se garantiza que el sistema cumpla con los requerimientos funcionales y no funcionales exigidos, se realicen las validaciones e impactos que tendrá al momento de su implementación con los diferentes softwares con los que interactúa y adicional prevenir posibles errores en el desarrollo de las actividades de los funcionarios.

Para realizar el proceso de pruebas de software es necesario contar con una formación académica la cual brinde las competencias necesarias para identificar cuáles son los tipos de pruebas que existe, sus utilidades y los momentos en los cuales se deben aplicar. Así como tener presente las diferentes metodologías que existen con el fin de analizar y determinar cuál es la mejor opción para el proyecto en el cual se está trabajando.

Debido a lo anterior se evidencia la necesidad de realizar una investigación donde se clarifiquen los conceptos sobre las pruebas de software, los tipos, los diferentes escenarios en donde aplicarlos y la importancia que tiene en el ámbito tecnológico este proceso, para así clarificar el proceso y demostrar a los actores involucrados como empresas, instituciones educativas, grupos de investigación, docentes y estudiantes la importancia del testing durante el ciclo de desarrollo de software.

VI. Conceptos Generales de Pruebas de Software

Al momento de realizar un desarrollo de software, tanto para un desarrollo nuevo como para una modificación de algún software existente, es necesario realizar validaciones de que el proceso cuenta con la calidad y entradas y salidas correctas esperadas por el usuario final.

Para poder determinar si es correcto se deben realizar pruebas de software en un ambiente controlado donde se ingresen las variables de entrada, el sistema realice el procedimiento para lo que fue diseñado y analizar la información saliente. A este proceso se le denomina prueba de software.

Para el proceso de pruebas son necesarios tres aspectos básicos:

- Entradas: Son las variables o información que necesita el sistema para iniciar su funcionamiento
- Procesos: Son los procedimientos internos con los que cuenta el sistema los cuales se encargaran de realizar todas las validaciones, estructuración, cálculos o funcionamiento para lo que fue diseñado el software
- Salidas: Es lo que debe entregar el software al momento de finalizar todos los procesos. Las salidas pueden ser de diversos tipos y formas, es decir, pueden ser datos, archivos, comunicación con otro sistema, entre otros.

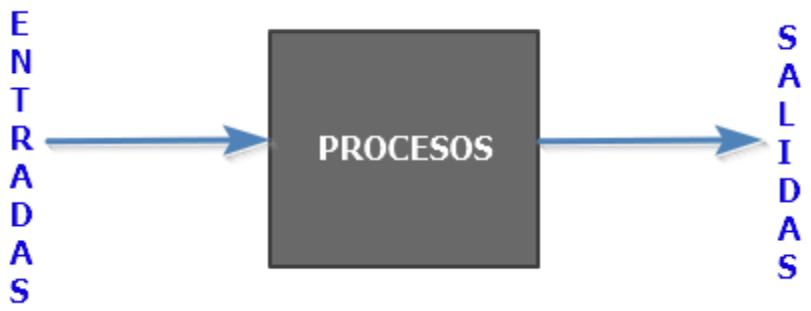

Ilustración 1 Fuente propia

Cuando el sistema después de realizar el proceso y mostrar las salidas no genera la información deseada se procede a realizar la notificación al desarrollador con toda la información referente al caso de prueba donde se le indica el procedimiento realizado paso a paso, la información ingresada y los resultados obtenidos con el fin de que sea revisado y corregido el sistema.

Al realizar este procedimiento se hace necesario volver a iniciar el ciclo completo de verificaciones.

VII. Tipos de Pruebas de Software

- Pruebas Software Funcionales

  Las pruebas funcionales son aquellas que se realizan para el comportamiento del sistema. Su objetivo principal es garantizar que las funcionalidades de los aplicativos realicen el proceso que deben realizar. Generalmente estas pruebas están descritas en las especificaciones de los requisitos o en los casos de uso. De igual forma también se pueden presentar en validaciones no descritas.

Para definir estas pruebas se debe tener claridad de las funciones o características del sistema y su interoperabilidad con los demás sistemas o aplicativos.

- Pruebas de Caja Negra

  Se consideran pruebas de caja negra a las realizadas al sistema de manera externa, es decir, se valida la parte externa y la relación con los demás sistemas sin validar el código línea a línea.

- Pruebas de Caja Blanca

  Se consideran pruebas de caja blanca a las pruebas que se realizan al código escrito línea a línea. Estas pruebas buscan revisar el funcionamiento lógico del desarrollo del sistema y detectar posibles errores desde las líneas directamente. Para realizar estas pruebas se utiliza una herramienta fundamental que se llama debug.

  El debug consiste en ir paso a paso por cada línea de código verificando lo que hace el sistema y mirando los valores que toman las variables con el fin de determinar el correcto funcionamiento o posibles puntos de errores.

- Pruebas Software no Funcionales

Este tipo de pruebas se enfocan en validar en cómo trabaja el sistema por lo tanto se enfocan en validaciones de rendimiento, carga, estrés, usabilidad, mantenibilidad, fiabilidad o portabilidad, entre otros.

- Pruebas Software de Regresión

  Estas pruebas se realizan sobre el sistema cuando este tuvo modificaciones para validar que con las correcciones realizadas no se hubiera dañado.

- Re-pruebas

  Las pruebas de regresión consisten en volver a probar un componente que presentó alguna inconsistencia y fue corregido para validar su correcto funcionamiento.

VIII. Aplicaciones para Pruebas de Software

A través del tiempo las pruebas de software se han vuelto importantes en el desarrollo del software lo que ha permitido que se creen departamentos y demás disciplinas enfocadas en la realización de estas y se amplíen los tipos de pruebas y demás procesos ocasionando una gran cantidad de actividades manuales repetitivas que deben hacer los funcionarios encargados de las pruebas.

Debido a lo anterior se han desarrollado diferentes herramientas que permitan automatizar de manera significativa los procesos de pruebas y logrando una mayor optimización del tiempo de los tester de software.

Como ejemplo de herramientas se explican las siguientes que son dentro del mercado de las más significativas:

- Selenium: Herramienta que permite crear casos de pruebas para aplicaciones web.
- JMeter: Herramienta permite realizar pruebas funcionales y de rendimiento para aplicaciones web.
- Testlink: Herramienta que permite administrar los casos de pruebas y se puede integrar con otros sistemas de seguimiento de bugs.

A continuación, se mencionan diferentes herramientas de software libre según su objetivo:

1. Herramientas de gestión de pruebas
    - Bugzilla Testopia
    - FitNesse
    - qaManager
    - qaBook
    - RTH (open source)
    - Salome-tmf
    - Squash TM
    - Test Environment Toolkit
    - TestLink

- Testitool
- XQual Studio
- Radi-testdir
- Data Generator

2. Herramientas para pruebas funcionales

- Selenium
- Soapui
- Watir (Pruebas de aplicaciones web en Ruby)
- WatiN (Pruebas de aplicaciones web en .Net)
- Capedit
- Canoo WebTest
- Solex
- Imprimatur
- SAMIE
- ITP
- WET
- WebInject

3. Herramientas para pruebas de carga y rendimiento

- FunkLoad
- FWPTT load testing

- loadUI
- jmeter

A continuación, se mencionan diferentes herramientas de software comercial según su objetivo:

1. Herramientas de gestión de pruebas
    - HP Quality Center/ALM
    - QA Complete
    - qaBook
    - T-Plan Professional
    - SMARTS
    - QAS.Test Case Studio
    - PractiTest
    - SpiraTest
    - TestLog
    - ApTest Manager
    - Zephyr

2. Herramientas para pruebas funcionales
    - QuickTest Pro
    - Rational Robot
    - Sahi
    - SoapTest

- Test Complete
- QA Wizard
- Squish
- vTest
- Internet Macros

3. Herramientas para pruebas de carga y rendimiento
    - HP LoadRunner
    - LoadStorm
    - NeoLoad
    - WebLOAD Professional
    - Forecast
    - ANTS – Advanced .NET Testing System
    - Webserver Stress Tool
    - Load Impact
    - Herramientas Todo en Uno

IX. Metodologías

El objetivo de las Metodología de pruebas es proporcionar a los analistas de pruebas, procedimientos y herramientas de fácil uso, poco densos y de gran efectividad a la hora de realizar pruebas.

Se debe tener en cuenta que en el proceso de desarrollo se cuenta con una metodología que tiene las características necesarias para aportar a las necesidades particulares del proyecto u organización para la que se crea, y que tiene como finalidad estandarizar, aumentar la trazabilidad y mejorar el resultado de las pruebas, y por tanto la calidad del software desarrollado.

A Continuación, se plantea una metodología de prueba que permita a los miembros del equipo de pruebas contar con un procedimiento claro y efectivo, para la realización de cada una.

Planeación de pruebas

1. Determinar la aplicación que va a ser sometida al proceso de pruebas: Esto debido a que todo desarrollo que se lleve a cabo no puede ser puesto a prueba por diversas razones (falta de recursos, disponibilidad de tiempo, poca criticidad, etc.). Por lo tanto, se hace necesario que la persona encargada de la implementación determine determinen que aplicación debe ser sometida al proceso de prueba y determinar el alcance de las pruebas a realizar.

2. Definir qué tipos de pruebas se realizarán a la aplicación previamente seleccionada. Esta actividad debe ser llevada a cabo por la persona encargada de Aseguramiento de Calidad.
3. Una vez se hayan determinado los tipos de pruebas que se van a realizar, es necesario seleccionar los módulos y características de la aplicación a probar.
4. Determinar que partes o módulos de la aplicación tienen más riesgo a fallas. Esto con el fin de dar prioridad a ciertos partes o módulos de la aplicación, que tienen más probabilidades de fallas, en el proceso de pruebas.
5. Definir claramente los criterios de aceptación.
6. Realizar un cronograma con todas las actividades consideradas dentro de todo el proceso de pruebas.

Preparación del ambiente de pruebas

1. Informar a la persona responsable de Aseguramiento de Calidad, por parte del Coordinador del equipo de desarrollo, que el desarrollo de la aplicación que va a hacer sometida al proceso de pruebas ha concluido.

El Coordinador de Desarrollo debe adjuntar toda la información necesaria para que pueda ser implementado el ambiente de pruebas de la aplicación, incluyendo:

- Requisitos funcionales y no funcionales
- Manual de usuario (Sí aplica)
- Casos de uso

- Diagrama de datos (especificando de donde deben obtenerse los datos para poblar la base de datos)
- Especificaciones técnicas:
    - Hardware y software necesarios
    - Servidor de aplicaciones
    - Motor de bases de datos
- Usuarios y contraseñas para acceder a las bases de datos.

2. Solicitar al Área de pruebas la implementación del ambiente de pruebas (Ver anexo B) para la aplicación a probar. Esta solicitud la realiza la persona encargada de Aseguramiento de la Calidad inmediatamente después de recibir la información enviada por el Coordinador del equipo de desarrollo.

    El ambiente de pruebas debe incluir:

    - Asignación de un servidor con las especificaciones técnicas requeridas.
    - Creación de los perfiles de usuario necesarios para tener acceso a todas las instancias del aplicativo.
    - Instalación del motor de la base de datos que alojará los datos para realizar las pruebas.
3. El Coordinador de pruebas debe informar cuando esté listo el ambiente de prueba.

A. Ejecución de pruebas de interfaz

Se realiza la descripción de los pasos que deben seguir los analistas al realizar pruebas sobre el interfaz de usuario.

1. Realizar una lista de pantallas de la aplicación y categorizarlas por tipo de usuario. Existen pantallas comunes a varios tipos de usuarios, por ejemplo, la pantalla de ingreso.
2. Seleccionar las pantallas que sean comunes a más usuarios, ordenarlas empezando por las que tengan en común más tipos de usuarios y terminando con aquellas propias de cada uno, esto se hace con el fin de agilizar el proceso de prueba.

   Se recomienda realizar un mapa jerárquico, abarcando todas las pantallas, que refleje la navegación por la aplicación. Realizar el ordenamiento es más fácil a partir del mapa.

3. Definir de los siguientes tipos de prueba aquellas que sean convenientes a aplicar para la pantalla seleccionada de la lista anterior, para posteriormente diligenciar las plantilla (Ver anexo D). Cada analista determina el número de pruebas a aplicar por pantalla.
   - Pruebas de link. Para probar un link es necesario dar clic en él y comprobar si está funcionando y re-direccionando al contenido apropiado.
   - Pruebas de formulario: Realizar pruebas de formulario incluye:
     - Asegurar que las etiquetas identifiquen correctamente los campos correspondientes.
     - Comprobar que los campos obligatorios están resaltados para una visualización correcta.
     - Los campos del formulario deben contener el tipo de dato apropiado con su respectiva amplitud.

- Los formularios deben establecer una restricción apropiada que impida al usuario ingresar cadenas de caracteres más largas de lo permitido, o en su defecto tipos de datos donde no correspondan.
- Todas las opciones en un menú desplegable deben tener un orden pertinente para el usuario.
- La función de "auto-relleno" de los navegadores no debe generar errores en el ingreso de los datos.
- La tecla tabuladora debe mover el cursor de manera secuencial entre los campos del formulario.
- Además, el analista por cada campo del formulario debe tener en cuenta las posibles condiciones de entrada, utilizando las técnicas de partición equivalente y análisis de valor límite, y de acuerdo ellas definir la cantidad y el valor de los datos a ingresar para realizar las pruebas de forma eficiente. Las posibles condiciones de entrada son:
  - El valor del campo está especificado dentro de un rango. Para realizar pruebas a este tipo debe ingresarse tres valores válidos, un valor dentro rango y los dos valores límite, y dos valores inválidos, es decir fuera del rango.
  - El valor del campo requiere un valor específico. Debe ingresarse el valor valido y dos inválidos, uno mayor y el otro menor que el valor aceptado.
  - El valor del campo especifica un miembro de un conjunto. Debe ingresarse un valor dentro del conjunto (valor valido) y otro por fuera de él.
  - El valor del campo especifica una variable booleana. Debe ingresarse un valor las dos posibles opciones de la variable.

- Pruebas scripts del lado del cliente: Realizar estas pruebas implica comprobar que los scripts de la pantalla realicen un chequeo sobre los datos ingresados en los formularios de dicha pantalla y, además, generen los mensajes de error pertinentes.
- Pruebas del contenido HTML Dinámico: Debe comprobarse que cada página que contenga contenido HTML dinámico lo presente de forma adecuada y pertinente.
- Pruebas de ventanas emergentes: Las pruebas de ventanas emergentes buscan lo siguiente: la ventana emergente debe estar dimensionada y posicionada apropiadamente; la ventana emergente no debe ocultar la ventana original de la aplicación; el diseño visual de las ventanas emergentes debe ser consistente con el diseño visual de toda la interfaz; y por último, las barras de desplazamiento y otros mecanismos de control utilizando para el funcionamiento de las ventanas emergentes, deben estar ubicadas y funcionado correctamente.
- Pruebas de "streaming": Las pruebas hechas a contenidos streaming deben demostrar que el contenido que se está transmitiendo sea presentado correctamente, además, pueda ser pausado y reiniciado sin ningún problema.

4. Realizar las pruebas a las pantallas seleccionadas con anterioridad y anotar los resultados en la plantilla de pruebas de interfaz (Ver anexo D). Una copia de dicha plantilla debe ser enviada al analista de desarrollo después de realizar las pruebas para que él depure los errores encontrados
5. El analista de desarrollo debe informar al analista de prueba sobre la depuración de los errores encontrados en el proceso de prueba (Ver anexo E). El analista de desarrollo debe

nombrar en las anotaciones los errores que no fueron depurados, así como también las posibles pantallas que pudieron verse afectadas por el proceso de depuración.

6. Realizar las pruebas a los errores corregidos y a todas aquellas pantallas que pudieron verse afectadas en el proceso de depuración. Las pantallas que pudieron verse afectadas por la depuración son recomendadas por el analista de desarrollo que realiza las correcciones.
7. En caso de encontrar algún error, bien sea nuevo o que haya sido informado con anterioridad al equipo de desarrollo, el analista de pruebas debe incluirlo dentro de una plantilla (Ver anexo D), incrementando el número de versión para ello, debe convocarse al responsable de Aseguramiento de la Calidad, al Coordinador de Desarrollo de Software y a los analistas, tanto de prueba como de desarrollo, a una reunión para definir el impacto que generan los errores encontrados en la última prueba. Es aconsejable cuantificar y priorizar el impacto entre uno y cinco, siendo el indicador de mínimo impacto uno, y cinco, el de mayores implicaciones y, por lo tanto, son estos últimos, los que deben depurarse antes de lanzar la aplicación en vivo.

   Los asistentes a la reunión están en libertad, de acuerdo con su experiencia y a las necesidades del proyecto, de definir las acciones que se deben tomar para solucionar los problemas mencionados anteriormente. El resultado de la reunión debe ser escrito en el formato de reuniones con el que cuenta el centro de informática (Ver anexo F).
8. Una vez el analista de desarrollo realice la depuración, el proceso debe volver al paso 5 de esta metodología y repetir el ciclo hasta que los miembros de la reunión consideren que el software que se está probando cumple con los requerimientos a nivel de interfaz.

B. Ejecución de pruebas de usabilidad

Las pruebas de usabilidad son una extensión de las pruebas de interfaz, ya que están directamente relacionadas con la apariencia, distribución y usabilidad de esta. Pero las pruebas son realizadas por un pequeño grupo de usuarios seleccionados para tal fin.

A continuación, se describen los pasos que se deben realizar en el proceso de pruebas de usabilidad:

1. se debe definir una categoría de las opciones presentadas para realizar este tipo de pruebas, ya que las preguntas de la encuesta que se le realizará al usuario (Ver anexo J) dependerán del conjunto de categorías seleccionado.

La selección de cada categoría depende de la aplicación a probar ya que no todas las categorías son evaluadas en las diferentes aplicaciones.

- Interactividad: se valida que los menús, botones, barras de desplazamiento, ayuda, y otros mecanismos de interactividad, son fáciles de usar y entender.
- Plantilla: Asegura que los mecanismos de navegación, el contenido y las funciones estén dispuestas de manera que los usuarios pueden encontrarlos rápidamente.
    - Legibilidad: Se valida que lo publicado en la aplicación tanto texto como la parte grafica se pueda comprender con facilidad
    - Estética: Asegura que la interfaz tenga un diseño (colores, figuras, etc.) agradable a las vistas de usuarios.

- Características de presentación: Garantiza que la aplicación utilice de manera óptima el tamaño y la resolución de la pantalla.
- Sensibilidad de tiempo: Determina si es necesario utilizar funciones o características asociadas con la utilización de tiempos.
- Personalización: determina si algunos componentes de la aplicación son personalizables de acuerdo con los gustos y necesidades de los usuarios.

2. Los usuarios finales que realicen las pruebas de usabilidad deben diligenciar la encuesta (ver anexo) seleccionando las preguntas que considere pertinentes de acuerdo con las características seleccionadas en el numeral anterior.
3. Se debe seleccionar un grupo de usuarios entre todos los participantes en el proceso de prueba, se sugiere sean más de 7 (o el número que considere la persona encargada del aseguramiento de la calidad, el número debe ser impar) para luego realizarles la encuesta.
4. Preparar a los usuarios seleccionados para el proceso de pruebas.
5. Dar un tiempo de uso adecuado para probar la aplicación.
6. Se entrega la encuesta entre los usuarios seleccionados.
7. Revisar las respuestas y sacar los resultados de cada pregunta de la encuesta de la siguiente manera:
    - Para preguntas con respuestas abiertas, el analista de pruebas debe resumirlas y sacar las conclusiones necesarias.
    - Para preguntas con una única respuesta tipo si/no se toma la opción que tuvo mayor uso.
    - Por cada pregunta se deben sumar los valores equivalentes de cada pregunta y luego se divide por el número de personas que contestaron la pregunta.

8. En caso de que hay cambios se debe enviar una copia de estos cambios al equipo de implementación, de ser necesario la creación de nuevos módulos se deben realizar las pruebas que se consideren necesarias.

C. Ejecución de pruebas de integridad

A continuación, se describen los pasos que se deben llevar a cabo para la realización de pruebas de integridad, hay que tener en cuenta que son pruebas de los métodos y procesos utilizados para acceder y gestionar las bases de datos, se debe garantizar que los procesos y las reglas de los datos funcionan como se espera y que, durante el acceso a la base de datos, los datos no se corrompan, sean borrados, modificados o creados de forma inesperada.

1. Seleccionar el módulo a probar de acuerdo los numerales 1 y 2 de las pruebas de interfaz.
2. Según los módulos seleccionados se debe determinar qué campos se deben probar, teniendo en cuenta que deben ser campos que escriban en la base de datos.
3. Probar los campos con valores de prueba, y diligenciar la plantilla correspondiente (Ver anexo G),
4. El analista de prueba debe informar los errores encontrados y enviarlos al analista de desarrollo
5. El analista de desarrollo debe informar al analista de pruebas sobre la corrección de los errores encontrados en el proceso de prueba, el analista de desarrollo debe enunciar los

errores que no fueron depurados y los posibles campos que pudieron sufrir alguna alteración en el proceso de depuración.

6. Se deben realizar pruebas a los errores corregidos y a todos los campos que fueron afectados.
7. Si se encontraron nuevos errores o los anteriores no fueron corregidos se debe enviar en una plantilla (ver anexo F) esta debe incrementar el número de versión para tener certeza que se realizó nuevamente la prueba.
8. Según los errores encontrados en la última versión, el equipo de calidad y el equipo de desarrollo y deben determinar si los errores afectan gravemente el desempeño de la aplicación, cuantificar la gravedad de los errores de 1 a 5, los que tengan la calificación más alta deben depurarse antes de la salida en vivo de la aplicación.
9. Cuando el analista de desarrollo corrija los errores, el proceso de pruebas en el paso 6 se debe volver a ejecutar para garantizar que se realizó el proceso de pruebas y la aplicación cuenta con los requisitos de calidad

D. Ejecución de pruebas de desempeño: carga

1. Definir los usuarios que tendrán acceso a la aplicación
2. Determinar la herramienta de prueba que se va a utilizar, se deben tener en cuenta los siguientes factores: sistema operativo en el que se van a realizar las pruebas, el costo y la cantidad de usuarios soportados por la herramienta.
3. Definir los casos de pruebas teniendo en cuenta las siguientes variables:

- Cantidad de usuarios virtuales que realizaran la prueba, se debe tener en cuenta que el número de usuario debe ser fijo
- tiempo total de duración de la prueba, este tiempo está dado en minutos
- Tiempo entre clics de los usuarios virtuales.
- Creación de perfil a cada usuario, se recomienda asignar los perfiles que tenga la aplicación para hacer pruebas correspondientes.

Debe tener en cuenta que, si la herramienta seleccionada lo permite, debe activar el "ip-spoofing", que asigna diferentes direcciones IP a los usuarios virtuales.

Organizar los casos de prueba definidos en una tabla así (también esta adjunta al Anexo K)

Número del caso de prueba	**Perfil**	**Número de usuarios virtuales**	**Duración de la prueba**	**Tiempo entre clics**
Número consecutivo para cada una de los casos de prueba	Perfil de usuario con el que se realizara la prueba	Cantidad de usuarios virtuales con los que se Cargara la aplicación	Tiempo total que durara la prueba, dada en minutos	Tiempo entre clics que dura un usuario virtual

Tabla 1 Ejemplo caso de prueba

Después de realizar las configuraciones adecuadas es necesario que almacene una secuencia de recorrido por la aplicación. Esta secuencia permitirá a la herramienta simular el comportamiento de los usuarios virtuales creados.

4. Realizar pruebas para cada uno de los diferentes casos creados, así garantizamos el correcto funcionamiento de la aplicación
5. De todos los resultados obtenidos en las pruebas por cada caso de prueba, tener en cuenta los siguientes y registrarlos en el respectivo formato (ver anexo k)
    - Páginas ejecutadas.
    - Páginas con error.
    - Clics ejecutados.
    - Clics con error.
    - Kilo Bytes enviados.
    - Kilo Bytes recibidos.
    - Tiempo de respuesta por página.
    - Porcentaje de errores HTTP.
    - Porcentaje de Timeouts.
    - Velocidad del usuario recibiendo.
    - Velocidad del usuario mandando.
    - Porcentaje total de errores.
6. Se deben presentar los resultados a las áreas de calidad y desarrollo y determinar el rendimiento de la aplicación y definir si se hacen necesarios ajustes de esta. El resultado de la reunión debe ser escrito en un documento (Ver anexo E) para registrar las decisiones tomadas.

Este proceso se repite hasta que las personas encargadas del proceso definan que la aplicación cumple con el rendimiento correcto.

E. Ejecución pruebas de desempeño: estrés

Se recomienda utilizar las mismas herramientas que se usaron en las pruebas de carga.

1. Definir los valores de las variables iníciales, en la tabla dispuesta la plantilla de stress (Ver Anexo L) necesarias para realizar las pruebas de stress, se deben tener en cuenta los siguientes aspectos:

    - Para este tipo de prueba es necesario que se configure el programa de forma que el número de usuarios virtuales crezca de manera incremental.
    - La duración total de la prueba debe ser igual o mayor que la duración fijada para las pruebas de carga.
    - El tiempo entre clics de los usuarios virtuales debe ser menor al tiempo fijado en las pruebas de carga.
    - Utilizar los mismos perfiles de las pruebas de carga para los usuarios virtuales.

    De nuevo, debe tener en cuenta que si la herramienta seleccionada lo permite debe activar el “ip-spoofing”, que asigna diferentes direcciones IP a los usuarios virtuales.

Una vez fijados los parámetros adecuados para ejecutar la prueba, utilizar la secuencia de navegación, creada para las pruebas de carga, para ejecutar la prueba de stress.

2. De los resultados obtenidos en la prueba, tener en cuenta los siguientes ítems por cada perfil de usuario definido:

    - Periodo de tiempo de ejecución
    - Número de usuarios virtuales en ese período de tiempo
    - Paginas ejecutadas
    - Paginas con error
    - Clics ejecutados
    - Clics con error
    - Kilo Bytes enviados
    - Kilo Bytes recibidos
    - Porcentaje de errores HTTP
    - Porcentaje de Timeouts
    - Velocidad del usuario recibiendo
    - Velocidad del usuario enviando
    - Porcentaje total de errores

    Registrar los resultados en la plantilla designada para este numeral (Ver Anexo L).

3. Se deben presentar los resultados a las áreas de calidad y desarrollo y determinar el rendimiento de la aplicación y definir si se hacen necesarios ajustes de esta. El resultado

de la reunión debe ser escrito en un documento (Ver anexo E) para registrar las decisiones tomadas.

Este proceso se repite hasta que las personas encargadas del proceso definan que la aplicación cumple con el rendimiento correcto.

F. Presentación de resultados

Cuando se habla de presentar los resultados, se deben incluir todos los errores encontrados en cada una de las pruebas realizadas, identificados por un código que los relacionen con el tipo de prueba donde fue descubierto; el estado es que este error se encuentra, es decir, si fue solucionado o está pendiente; la prioridad de solución, en caso de que esté pendiente de solucionar; y observaciones sobre el error (Ver anexo H).

El último paso después de asegurarnos que nuestra aplicación esta lista para salir a producción es informar por medio de una carta al líder de desarrollo. (Ver anexo I).

Conclusiones de la metodología de prueba

En este apartado el líder del área de calidad debe informar sobre el proceso de pruebas, retroalimentar al su equipo sobre los hallazgos que se encontraron y revisar si se la metodología es exitosa o corregir los aspectos a mejorar.

X. Estándares Internacionales

Para realizar las pruebas de software se han determinado estándares que permiten definir reglas y lineamientos para dicho proceso.

Dentro de los estándares y normas encontramos los siguientes:

- Estándares BSI
  - BS 7925-1, SW Testing: Part 1-Vocabulary
  - BS 7925-2, SW Testing: Part 2-Software Component Testing
- Estándares IEEE testing:
  - IEEE Std 829, Software Test Documentation
  - IEEE Std 1008, Software Unit Testing
- Estándares sectoriales
- Cuerpo de conocimiento y metodologías, p.e.
  - ISTQB
  - TMap Next®
- Otros estándares de tipo general: ISO/IEC 12207, 15289
- OEEE & BSI Standards
  - IEEE 829: documentación (proceso implícito)
  - IEEE 1008: pruebas unitarias
  - BS 7925-1 & 2: pruebas unitarias

XI. Conclusiones

En el desarrollo de esta investigación se definen los diferentes tipos de pruebas, así como las metodologías y las aplicaciones necesarias para su desarrollo y la importancia que tienen en el ciclo de vida de desarrollo e implementación de un proyecto de software.

Es importante entender el avance de las organizaciones consumidoras de software, teniendo en cuenta la exigencia por parte de los clientes sobre la calidad del software, así como el incremento de proveedores de servicios informáticos, han llevado a los usuarios de los productos de software a convertirse en parte importante del proceso de desarrollo de software sacando máximo provecho de la aplicaciones, retroalimentando sobre el desempeño, y sobre todo forzando a los implementadores a desarrollar aplicativos más eficientes, con mayor funcionalidad, amigables y sobre todo, productos de mejor calidad.

En esta investigación ese encuenttan las falencias que tiene la universidad en esta área tan importante en el proceso de implementación, por ello fue necesario desarrollar un manual sobre pruebas de software enmarcando las pruebas dentro de una metodología ajustada a las necesidades particulares de facilidad y eficacia que requiera la aplicación. Este trabajo se realiza con el fin de exponer la necesidad de la realizar las pruebas a las aplicaciones desarrolladas para que estas alcancen un nivel de calidad competitivo respecto a los estándares mundiales, cumpliendo y satisfaciendo las expectativas de los clientes y usuarios finales de las aplicaciones

La descripción propuesta sobre las pruebas es un paso importante hacia el mejoramiento del contenido de las materias de ingeniería de software, ya que se muestra toda la información

necesaria para la realización de las pruebas en el ciclo de desarrollo e implementación. sin embargo, la metodología no soporta todo el proceso de desarrollo de software.

Con este trabajo queremos acceder a nuestro grado de Ingenieras de Sistemas y Computación

Bibliografía

- https://www.panel.es/blog/software-qa-cuales-son-los-tipos-de-pruebas-software/
- http://www.tamushi.com/2014/05/diferencia-error-defecto-y-fallo/
- https://www.definicionabc.com/general/sistema.php
- PRESSMAN, Roger S. "Ingeniería del software: Un enfoque práctico", 7a edición,Mc Graw Hill. 2006
- https://www.globetesting.com/pruebas-de-integridad-de-base-de-datos/
- https://testeandosoftware.com/las-mejores-herramientas-para-realizar-pruebas-de-software/
- https://testeandosoftware.com/10-herramientas-para-pruebas-de-software-ii/
- http://www.javiergarzas.com/2012/03/herramientas-para-pruebas-software.html
- https://testingbaires.com/16-herramientas-open-source-testers/
- http://in2test.lsi.uniovi.es/gt26/presentations/ISO29119-Presentacion-GT26-20140618.pdf
- http://www.iingen.unam.mx/esmx/Publicaciones/GacetaElectronica/Agosto2016/Paginas/Erroresdesoftware.aspx

Anexos

Anexo A Informe De Desarrollo Concluido

Descripción: documento para informar sobre una aplicación ya terminada, y se solicita que se prepare para pruebas

Señor

Nombre completo

Líder de Aseguramiento de Calidad

ASUNTO: Informe de concluido el desarrollo de la aplicación *Nombre de la aplicación*.

Por medio de la presente se informa que se ha concluido el proceso de desarrollo de la aplicación "*Nombre de la aplicación"*. A partir de este momento pueden implementar el ambiente de prueba, adjunto la información necesaria.

A continuación, se listan los posibles elementos que contiene la documentación necesaria para implementar el ambiente de pruebas:

- Requisitos funcionales y no funcionales
- Manual de usuario (Sí aplica)
- Casos de uso
- Especificaciones técnicas:
    i. Hardware y software necesarios
    ii. Servidor de aplicaciones
    iii. Motor de bases de datos
    iv. Ubicación de los datos

Anotaciones:

Cualquier anotación que considere pertinente

_____

Líder de Desarrollo de Software

Anexo B Solicitud De Ambiente De Pruebas

Descripción: Formato pasa solicitar la preparación del ambiente de pruebas, se incluye que los requisitos técnicos del sistemas, asi como la creación de perfiles necesarios para probar la aplicación.

Señor

Nombre completo

Coordinador área de pruebas

ASUNTO: Solicitud de implantación del ambiente de pruebas para la aplicación "*Nombre de la aplicación"* y población de las bases de datos.

Por medio de la presente se solicita al Área de pruebas gestionar la implementación del ambiente de pruebas para la aplicación *"Nombre de la aplicación"*.

Se adjunta la documentación y especificaciones técnicas necesarias para el correcto funcionamiento de la misma.

El ambiente de pruebas debe incluir:

- Asignación de un servidor con las especificaciones técnicas requeridas.
- Creación de los perfiles de usuario, necesarios para tener acceso a todas las instancias del aplicativo.
- Instalación del motor de la base de datos que alojará los datos para realizar las pruebas.

Anotaciones:

Informar, cuando esté listo el ambiente de prueba al Coordinador de Aseguramiento de Calidad.

_____

Líder de Aseguramiento de Calidad

Anexo C . Informe De Ambiente De Pruebas

Descripción: documento para informar que el ambiente de pruebas ya esta listo, es decir, ya se puede empezar a probar la aplicación.

Señor

Nombre completo

Líder de Aseguramiento de Calidad

ASUNTO: Informe de implementación del ambiente de pruebas para la aplicación "*Nombre de la aplicación"*.

Por medio de la presente se informa al Área de Aseguramiento de Calidad que el ambiente de pruebas para la aplicación "*Nombre de la aplicación"* se encuentra listo y pueden empezar a realizar las pruebas convenientes.

Anotaciones:

En caso que el Área de pruebas no pueda poblar alguna tabla o base de datos, debe informar al Equipo de Aseguramiento de Calidad para que aquellas que no pudieron ser

pobladas, sean pobladas por este último equipo

_____

Líder de Aseguramiento de Calidad

Anexo D Plantilla de Interfaz

Descripción: plantilla donde se registran las pruebas de interfaz, el tipo de pantalla que se probó y si se acepta o rechaza

Nombre de la interfaz: *Nombre de la interfaz probada*

Descripción: *Describir el usuario al que pertenece la interfaz y la función de la pantalla probada.*

Número	**Rol/Usuario**	**Campo**	**Tipo**	**Prueba realizada**	**Datos de prueba**	**A**	**R**	**Comentarios**
Consecutivo del caso de prueba	*Rol y usuario con el cual se ejecuto la prueba*	*Nombre del campo probado*	*De dato aceptado por el campo*	*Nombre del tipo de prueba*	*Datos ingresados para pruebas.*	*Marcar con "X" si lo acepta*	*Marcar con "X" si lo rechaza*	

_____ _____

Analista de prueba Analista de desarrollo

Se adjunta impresión de la panta

Anexo E Informe De Corrección

Descripción: formato donde se informa de la corrección de errores al analista de pruebas, es decir, que puede volver a probar la aplicación para validar la corrección

Señor

Nombre completo

Analista de prueba

ASUNTO: Informe de corrección de errores de la plantilla de pruebas de código *"código de la plantilla depurada".*

Por medio de la presente me permito informarle que se ha realizado la depuración de los errores presentados en la plantilla de código *"Código de la plantilla"*.

Anotaciones:

Describir en anotaciones los errores que no pudieron ser solucionados, incluyendo la causa para no depurarlos y la acción a tomar por cada uno ellos.

En caso que considere que alguna pantalla pude verse afectada por el proceso de depurado debe nombrarla en este espacio.

_____

Analista de desarrollo

Anexo F Reuniones

Descripción: Acta donde queda registrada la reunión de revisión de avances y finalización de los proyectos, en esta acta se debe registrar el nombre del proyecto, la fecha, hora y lugar de la reunión, asistentes, invitados, ausentes, los acuerdos, los responsables de cada actividad y la fecha limite de solución.

Proyecto:

Fecha:

Hora:

Lugar:

Asistentes

Nombre	Rol / Cargo	Dependencia

Invitados

Nombre	Rol / Cargo	Dependencia

Ausentes

Nombre	Rol / Cargo	Dependencia

Asunto

Desarrollo

Compromisos

Descripción	Responsable	Fecha Límite

Anexo G Plantilla De Integridad

Descripción: plantilla donde se registran las pruebas de integridad, aquí se validan los campos y que si se realice el almacenamiento exitoso en la base de datos.

Nombre de la interfaz: *Nombre de la interfaz probada*

Descripción: *Describir el usuario al que pertenece la interfaz y la función de la pantalla probada.*

Número	**Rol/Usuario**	**Campo**	**Tipo**	**B.D. y Tabla**	**Prueba realizada**	**Datos de prueba**	**A**	**R**	**Comentarios**
Consecutivo del caso de prueba	*Rol y usuario con el cual se ejecutó la prueba*	*Nombre del campo probado*	De dato aceptado por el campo	Nombre de la B.D. y tabla en la que escribe	Nombre del tipo de prueba	Datos ingresados para pruebas.	Marcar con "X" si lo acepta	Marcar con "X" si lo rechaza	Si se presenta un error de escritura debe explicarlo acá.

_____ _____

Analista de prueba Analista de desarrollo.

Anexo H Plantilla Final

Descripción: plantilla donde se registra la finalización de las pruebas, con los errores que se registraros y las observaciones sobre el mismo.

Código	**Tipo Error**	**Nombre Error**	**Estado**	**Prioridad de Solución**	**Observaciones**
Código del error compuesto por Código de la Plantilla y el consecutivo del caso de prueba, por ejemplo PInterfaz0001-1	*Tipo de error*	*Nombre del error*	*Estado en el que está error, corregido o pendiente*	*Número asignado por el coordinador de Aseguramiento de calidad, siempre entre 1 y 5, siendo 1 el más bajo y 5 el más alto*	*Observaciones pertinentes sobre el error.*

_____ _____

Analista de prueba Analista de desarrollo

Anexo I Informe De Conclusión De Pruebas

Señor

Nombre completo

de Desarrollo de Software

ASUNTO: Informe de concluido de las pruebas a la aplicación "*Nombre de la aplicación*"

Por medio de la presente se informa que se ha concluido el proceso prueba de la aplicación "*Nombre de la aplicación"*. A partir de este momento pueden sacar la aplicación a producción.

_____

Coordinador de Desarrollo de Software

Anexo J Encuesta Pruebas De Usabilidad

Por favor califique cada uno de las siguientes características, utilizando la escala de calificación que sigue:

Calificación 5: Si está totalmente de satisfecho
Calificación 4: Si está satisfecho
Calificación 3: Si está medianamente satisfecho
Calificación 2: Si está insatisfecho
Calificación 1: Si está totalmente insatisfecho

Id.	**Pregunta**	**Respuesta**						
1.	La facilidad de uso de:		**5**	**4**	**3**	**2**	**1**	**N/A**
		1.1. Menús						
		1.2. Botones						
2.	La disposición de:		**5**	**4**	**3**	**2**	**1**	**N/A**
		1.1. Menús						
		1.2. Contenido						
		1.3. Barras de desplazamiento						
		1.4. Funciones						
		1.5. Gráficos						
3.	La facilidad de lectura de:		**5**	**4**	**3**	**2**	**1**	**N/A**
		1.1. Contenido escrito						
		1.2. Contenido gráfico						
4.	El estética de:		**5**	**4**	**3**	**2**	**1**	**N/A**
		1.1. Colores						
		1.2. Figuras						
		1.3. Tablas						
		1.4. Botones						
		1.5. Letra						

Id.	Pregunta	Respuesta					
5.		5	4	3	2	1	N/A
	La distribución de la aplicación en la pantalla						
6.		5	4	3	2	1	N/A
	El tiempo de respuesta de la aplicación						

Id.	Pregunta	Respuesta
7.	Considera usted que la aplicación debe permitir personalizarse	S/N ___

Id.	Pregunta	Respuesta
8.	¿Qué piensa usted que falta en la aplicación?	
9.	Comentarios y sugerencias	
10.	Nombres y apellidos	
11.	Email	

Anexo K Plantilla De Carga

Nombre De La Aplicación: *Nombre De La Aplicación Probada*

Nombre De La Herramienta: *Nombre De La Herramienta Usada Para Realizar Las Pruebas De Carga.*

Número Del Caso De Uso	Perfil	Número De Usuarios Virtuales	Duración De La Prueba	Tiempo Entre Clics
Número Consecutivo Para Cada Una De Los Casos De Prueba	Perfil De Usuario Con El Que Se Realizara La Prueba	Cantidad de usuarios virtuales con los se cargar la aplicación	Tiempo Total Que Durara La Prueba, Dada En Minutos	Tiempo Entre Clics Que Dura Un Usuario Virtual

Número de caso prueba	**Paginas ejecutadas/ Paginas con error**	**Clics ejecutados /Clics con error**	**Tiempo de respuesta**	**Kbytes enviados / Kbytes recibidos**	**Porcentaje de errores HTTP**	**Porcentaje de Timeouts**	**Velocidad del usuario recibiendo / Velocidad del usuario mandando**	**Porcentaje total de errores**
Consecutivo del caso de prueba	*Número de páginas ejecutadas y número de páginas que presentaron error*	*Número de clics ejecutados y número De clics que presentaron error*	*La página donde se presentó el error*	*Numero de Kbytes enviados y Numero de Kbytes recibidos por perfil*	*Porcentaje de errores HTTP presentados en la prueba*	*Porcentaje de Timeouts presentados en la prueba*	*Velocidad del usuario tanto recibiendo como mandando*	*Porcentaje total de errores presentados en las pruebas.*

_____ _____

Analista de prueba Analista de desarrollo

Anexo L Plantilla De Stress

Nombre de la aplicación: *Nombre de la aplicación probada*

Nombre de la herramienta: *Nombre de la herramienta usada para realizar las pruebas de carga.*

Rango de usuarios virtuales	**Frecuencia de usuario virtuales**	**Duración de la prueba**	**Tiempo entre clics**
Rango de usuarios virtuales con los que se cargara la aplicación periódicamente	*Frecuencia con la que irán ingresando los usuarios virtuales a la aplicación. Esta frecuencia está dada en Usuarios/Segundos*	*Tiempo total que durara la prueba, dada en minutos*	*Tiempo entre clics que dura un usuario virtual*

Perfil de usuario	**Periodo de tiempo**	**Número de usuarios virtuales**	**Paginas ejecutada s / Paginas con error**	**Clics ejecutado s / Clics con error**	**Kbytes enviados / Kbytes recibido s**	**Porcentaje de errores HTTP**	**Porcentaje de Timeouts**	**Velocidad del usuario recibiendo / Velocidad del usuario envidando**	**Porcentaje total de errores**
Perfil de usuario en donde se presentó el error	*Periodo de tiempo en el que se presentaron los errores*	*Número de usuarios virtuales que había en el periodo de tiempo*	*Número de páginas ejecutadas y número de páginas que presentaron error en el periodo de tiempo*	*Número de clics ejecutados y número de clics que presentaron errores en el periodo de tiempo*	*Número de Kbytes enviados y Numero de Kbytes recibidos en el periodo de tiempo*	*Porcentaje de errores HTTP presentado s en el periodo de tiempo*	*Porcentaje de Timeouts presentado s en el periodo de tiempo*	*Velocidad del usuario tanto recibiendo como mandando en el periodo de tiempo*	*Porcentaje total de errores presentados en las pruebas en el periodo de tiempo*

_____ _____

Analista de prueba Analista de desarrollo